





SISTEMA INTEGRADO DE GESTIÓN DE EXPEDIENTES MODULAR (SIGM)

MANUAL DESARROLLADOR

ACCIONES MULTIENTIDAD

SIGM v3



Administración Local Soluciones

Control de versiones

Versión	Fecha aprobación	Cambio producido	Autor
1.0	Octubre 2012	Versión inicial	IECISA





ÍNDICE





1 Introducción

1.1 *Visión general del sistema*

AL SIGM es la plataforma de Tramitación Electrónica del MINETUR, solución integral para la tramitación electrónica de los procedimientos administrativos, que fomenta la interoperabilidad entre administraciones mediante su adaptación a estándares de comunicación así como la reutilización de recursos e información pública.

1.2 *Finalidad del documento*

El presente documento contiene la información necesaria para implementar acciones multientidad sobre entidades. Para revisar el proceso de ejecución de éstas acciones multientidad (cómo configurarlas y posteriormente ejecutarlas) se puede consultar la sección dedicada a acciones multientidad del documento:

SGM_*_*_ Manual de Usuario Administración Entidades

1.3 *Definiciones y Abreviaturas*

A continuación se expone una tabla con los diferentes acrónimos y abreviaturas utilizados a lo largo del documento, con su correspondiente definición.

Acrónimo / Abreviatura	Definición
MINETUR	Ministerio de Industria, Energía y Turismo
IECISA	Informática El Corte Inglés S.A.
SIGM	Sistema Integrado de Gestión de Expedientes Modular
AL	Administración Local

2 Desarrollo de acciones multientidad

2.1 *Consideraciones generales*

El desarrollo de una acción multientidad comprende la creación de dos clases Java; una configuradora que se encargará de definir los pasos de configuración que requiere la acción (siempre dentro de unos predefinidos) y por otra parte una acción ejecutora que obtendrá los valores recogidos por la acción configuradora y realizará las operaciones para las que está destinada.

Las acciones se deben implementar en el proyecto SIGEM_AdministracionWeb y para seguir el mismo criterio que las ya implementadas se colocarán en el paquete ieci.tecdoc.sgm.admsistema.action.accionesmultientidad, para las acciones configuradoras y ieci.tecdoc.sgm.admsistema.proceso.accionesmultientidad para las acciones ejecutoras.

2.2 *Acciones configuradoras*

Una acción configuradora es una clase Java que permite definir los pasos de configuración necesarios para una acción multientidad. Los pasos posibles serían los siguientes:

- Selección de entidad origen
- Selección de entidad destino
- Selección de entidades (cuando no hay una entidad origen ni destino)
- Selección de fichero
- Selección de opción de configuración (cuando la acción requieres opciones de configuración adicionales)
- Resumen de configuración
- Ejecución de la acción

La acción de configuración debe implementar el interface **IConfiguracionAccionMultientidad**:

```
package ieci.tecdoc.sgm.admsistema.action.accionesmultientidad;
```

```
import ieci.tecdoc.sgm.admsistema.form.AccionMultientidadForm;
import ieci.tecdoc.sgm.admsistema.vo.AccionMultientidadVO;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.ActionMapping;

/**
 * Interface para las action de configuracion de acciones
 * de multientidad
 * @author IECISA
 *
 */
public interface IConfiguracionAccionMultientidad {

  /**
   * Devuelve el siguiente paso a partir del formulario actual y de los datos de la peticion
   * @param mapping
   * @param form
   * @param request
   * @param response
   * @param accionMultientidadVO VO para obtener los valores de todo el proceso de ejecucion de accion de multientidad
   * @return siguiente paso a partir del formulario actual y de los datos de la peticion
   */
  public String executeConfigAction(ActionMapping mapping, AccionMultientidadForm form,
     HttpServletRequest request, HttpServletResponse response, AccionMultientidadVO accionMultientidadVO);
}
```

Es recomendable que la acción de configuración extienda la clase **ConfiguracionAccionMultientidadBaseAccion** ya que ésta última proporciona métodos de utilidad para el desarrollo de la acción.

El método **executeConfigAction** tiene los siguientes parámetros:

- mapping: ActionMapping de Struts.
- form: Formulario de Struts
- request: Petición actual

- response: Respuesta
- accionMultientidadVO: VO con los datos de configuración de la acción.

Como respuesta devuelve el nombre del paso que se debe ejecutar a continuación, siempre basándose en los valores definidos en la clase ConfiguracionAccionMultientidadConstants:

- PASO_SELECCION_ENTIDADES
- PASO_SELECCION_ENTIDADES_ORIGEN
- PASO_SELECCION_ENTIDADES_DESTINO
- PASO_SELECCION_FICHERO
- PASO_SELECCION_OPCION
- PASO_RESUMEN_CONFIGURACION
- PASO_EJECUCION_CONFIGURACION

2.2.1 Descripción de AccionMultientidadForm

La clase AccionMultientidadForm actúa de formulario de Struts y permite almacenar los siguientes valores:

- idAccion: identificador de la acción actual
- nombreAccion: nombre de la acción actual
- entidades: array de entidades seleccionadas
- opcionesConfiguracion: array de opciones de configuración
- claseConfiguradora: clase configuradora de la acción
- claseEjecutora: clase ejecutora de la acción
- opcion: opción seleccionada en pantalla
- fichero: fichero seleccionado en pantalla
- ficheroTemporal: fichero temporal almacenado (cuando se selecciona un fichero)
- paso: paso actual de configuración
- resumenConfiguracion: cadena que representa el resumen de la configuración

2.2.2 Descripción de AccionMultientidadVO

La clase AccionMultientidadVO actúa de Value Object para almacenar los valores de configuración obtenidos en cada uno de los pasos de la acción configuradora. Los valores que permite almacenar son los siguientes:

- idAccion: identificador de la acción actual
- nombreAccion: nombre de la acción actual
- entidades: array de entidades seleccionadas
- entidadesOrigen: array de entidades origen seleccionadas
- entidadesDestino: array de entidades destino seleccionadas
- claseConfiguradora: clase configuradora de la acción
- claseEjecutora: clase ejecutora de la acción
- opcionesConfiguracion: array de opciones de configuración
- opcion: opción seleccionada en pantalla
- ficheroTemporal: fichero temporal almacenado (cuando se selecciona un fichero)
- nombreFicheroTemporal: nombre del fichero temporal almacenado (cuando se selecciona un fichero)

2.2.1 Descripción de OpcionConfiguracionVO

La clase OpcionConfiguracionVO permite describir una opción de configuración de una acción configuradora:

- id: identificador de la opción de configuración.
- label: etiqueta de la opción de configuración.

2.2.2 Ejemplo de acción de configuración

```
package ieci.tecdoc.sgm.admsistema.action.accionesmultientidad;

import ieci.tecdoc.sgm.admsistema.form.AccionMultientidadForm;
import ieci.tecdoc.sgm.admsistema.vo.AccionMultientidadVO;
import ieci.tecdoc.sgm.admsistema.vo.OpcionConfiguracionVO;

import java.util.ResourceBundle;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.lang.StringUtils;
import org.apache.log4j.Logger;
import org.apache.struts.action.ActionMapping;
```

```
public class CompararImportarReglasAccionConfiguracion extends
ConfiguracionAccionMultientidadBaseAccion implements
    IConfiguracionAccionMultientidad {

  public String executeConfigAction(ActionMapping mapping,
      AccionMultientidadForm form, HttpServletRequest request,
      HttpServletResponse response, AccionMultientidadVO accionMultientidadVO) {
    String pasoActual = form.getPaso();
    try {
      if (StringUtils.isEmpty(pasoActual)){
        return ConfiguracionAccionMultientidadConstants.PASO_SELECCION_ENTIDADES_ORIGEN;
      } else if
(ConfiguracionAccionMultientidadConstants.PASO_SELECCION_ENTIDADES_ORIGEN.equals(pasoActual)){
        return ConfiguracionAccionMultientidadConstants.PASO_SELECCION_ENTIDADES_DESTINO;
      } else if
(ConfiguracionAccionMultientidadConstants.PASO_SELECCION_ENTIDADES_DESTINO.equals(pasoActual)){
        ResourceBundle rb =
ResourceBundle.getBundle("ieci/tecdoc/sgm/admsistema/resources/AdministracionMessage",
            request.getLocale());
        OpcionConfiguracionVO[] opcionesConfiguracion = new OpcionConfiguracionVO[2];
        opcionesConfiguracion[0] = new OpcionConfiguracionVO(ID_COMPARAR,
            rb.getString("acciones.multientidad.accion.comparar.importar.reglas.opcion.comparar"));
        opcionesConfiguracion[1] = new OpcionConfiguracionVO(ID_IMPORTAR,
            rb.getString("acciones.multientidad.accion.comparar.importar.reglas.opcion.importar"));
        form.setOpcionesConfiguracion(opcionesConfiguracion);
        return ConfiguracionAccionMultientidadConstants.PASO_SELECCION_OPCION;
      } else if (ConfiguracionAccionMultientidadConstants.PASO_SELECCION_OPCION.equals(pasoActual)){
        form.setResumenConfiguracion(generateResumenConfiguracion(accionMultientidadVO, request));
        return ConfiguracionAccionMultientidadConstants.PASO_RESUMEN_CONFIGURACION;
      } else if
(ConfiguracionAccionMultientidadConstants.PASO_RESUMEN_CONFIGURACION.equals(pasoActual)){
        return ConfiguracionAccionMultientidadConstants.PASO_EJECUCION_CONFIGURACION;
      }
      return null;
    } catch (Exception e) {
      logger.error(e);
      return null;
    }
  }

  public static final String ID_COMPARAR = "1";
  public static final String ID_IMPORTAR = "2";
```

```
  private static final Logger logger = Logger.getLogger(CompararImportarReglasAccionConfiguracion.class);

}
```

El método executeConfigAction permite definir el flujo de ejecución de la acción de configuración, y se debe encargar de establecer en el formulario las opciones de configuración necesarias (en el caso de que la acción necesite alguna) y el resumen de configuración (si se desea mostrar la pantalla de resumen).

2.3 *Acción ejecutora*

Las acciones ejecutoras simplemente se encargan de obtener los valores de configuración de la acción configuradora y realizar las operaciones necesarias para las que está destinada. Tienen que implementar el interfaz **IProcessManager**:

```
package ieci.tecdoc.sgm.admsistema.proceso;

import java.util.Map;

/**
 * @author Iecisa
 * @version $Revision$
 *
 */
public interface IProcessManager {

  /**
   * Realiza la ejecución de un proceso
   * @param options Parámetros para el proceso
   * @return true si el proceso se ha realizado con éxito.
   * @throws Exception si ocurre algún error.
   */
  public boolean execute(Map options) throws Exception;

}
```

El método execute permite incluir la lógica de ejecución de la acción, y devuelve un booleano indicando si la acción ejecutora se ejecutó correctamente (true) o no (false). Es recomendable que las clases que implementen éste interface extiendan de la clase




AccionEjecucionBase ya que proporciona métodos de utilidad para la ejecución de la acción.

A partir del parámetro options del método execute podemos obtener los siguientes valores:

- AccionMultientidadVO: permite obtener todos los valores de configuración almacenados por la clase configuradora. Lo podemos obtener mediante la siguiente sentencia:

```
AccionMultientidadVO accionMultientidadVO = (AccionMultientidadVO)
options.get(EjecutarAccion.PARAM_ACCION_MULTIENTIDAD_VO);
```

- Sesion: permite obtener los datos del usuario conectado y entidad. Lo podemos obtener mediante la siguiente sentencia:

```
ieci.tecdoc.sgm.sesiones.administrador.ws.client.Sesion sesion =
(ieci.tecdoc.sgm.sesiones.administrador.ws.client.Sesion)
options.get(EjecutarAccion.PARAM_SESION_APP_ADMINISTRACION);
```

- Nombre de la clase ejecutora: la podemos obtener mediante la siguiente sentencia:

```
String nombreClaseEjecutora = options.get(EjecutarAccion.
PARAM_ACCION_MULTIENTIDAD_NOMBRE_CLASE_EJECUTORA);
```

A continuación se puede ver un ejemplo de una acción que simplemente imprime algunos valores obtenidos de options:

```
package ieci.tecdoc.sgm.admsistema.proceso.accionmultientidad;

import ieci.tecdoc.sgm.admsistema.vo.AccionMultientidadVO;
import ieci.tecdoc.sgm.sesiones.administrador.ws.client.Sesion;

import java.util.Map;

public class PruebaAccionEjecucion extends AccionEjecucionBase {

		public boolean execute(Map options) throws Exception {
```

```
                        Sesion sesion = (Sesion) options.get(

            EjecutarAccion.PARAM_SESION_APP_ADMINISTRACION);
                        System.out.println("Entidad: "+sesion.getIdEntidad());
                        System.out.println("Usuario: "+sesion.getUsuario());

                        AccionMultientidadVO    accionMultientidadVO    =    (AccionMultientidadVO)
options.get(

            EjecutarAccion.PARAM_ACCION_MULTIENTIDAD_VO);
                        String [] entidades = accionMultientidadVO.getEntidades();
                        if (entidades!=null){
                                    for (int i=0;i<entidades.length;i++){
                                                System.out.println("Entidad: "+entidades[i]);
                                    }
                        }
                        System.out.println("Opcion                                        configuración:
"+accionMultientidadVO.getOpcion());

                        String claseEjecutora = (String) options.get(

            EjecutarAccion.PARAM_ACCION_MULTIENTIDAD_NOMBRE_CLASE_EJECUTORA);
                        System.out.println("Clase ejecutora: "+claseEjecutora);
                        return true;
            }

}
```

2.4 *Activación de la acción multientidad*

Una vez que se ha implementado tanto la acción configuradora como la ejecutora y se ha incluido las dos clases en el war de SIGEM_AdministracionWeb habría que dar de alta la nueva acción multientidad en el esquema de base de datos sigemAdmin mediante la sentencia:

```
INSERT INTO sgm_adm_acciones (id, nombre, clase_config, clase_exec, info_adicional) VALUES (ID_ACCION,
'NOMBRE_ACCION',
'ieci.tecdoc.sgm.admsistema.action.accionesmultientidad.CompararImportarReglasAccionConfiguracion',
'ieci.tecdoc.sgm.admsistema.proceso.accionmultientidad.CompararImportarReglasAccionEjecucion', NULL);
```

- ID: identificador de la acción, es único para la tabla sgm_adm_acciones

- NOMBRE_ACCION: nombre de la acción
- CLASE_CONFIGURADORA: nombre calificado completo de la clase configuradora, por ejemplo ieci.tecdoc.sgm.admsistema.action.accionesmultientidad.PruebaConfiguracion
- CLASE_EJECUTORA: nombre calificado completo de la clase configuradora, por ejemplo ieci.tecdoc.sgm.admsistema.proceso.accionesmultientidad.PruebaEjecucion

Con ésta inserción la clase ya aparecería entre las acciones multientidad en la Administración de entidades.